

# Nucleus NET PPP Driver

## Reference Manual



0001012-002

Copyright (c) 1999  
Accelerated Technology, Inc.  
720 Oak Circle Dr. E.  
Mobile, AL 36609  
(334) 661-5770



### Related Documentation

**Nucleus PLUS Reference Manual**, by Accelerated Technology, describes the operation and usage of the Nucleus PLUS kernel.

**Nucleus PLUS Internals**, by Accelerated Technology, describes, in considerable detail, the implementation of the Nucleus PLUS kernel.

**Nucleus NET Reference Manual**, by Accelerated Technology, describes the operation and usage of the Nucleus NET.

### Style and Symbol Conventions

Program listings, program examples, filenames, menu items/buttons and interactive displays are each shown in a special font.

Program listings and program examples - *Courier New*

Filenames - *COURIER NEW, ALL CAPS*

Interactive Command Lines - ***Courier New, Bold***

Menu Items/Buttons – *Times New Roman Italic*

### Trademarks

MS-DOS is a trademark of Microsoft Corporation

UNIX is a trademark of X/Open

IBM PC is a trademark of International Business Machines, Inc.

### Additional Assistance

For additional assistance, please contact us at the following:

Accelerated Technology  
720 Oak Circle Drive, East  
Mobile, AL 36609  
800-468-6853  
334-661-5770  
334-661-5788 (fax)

[support@atinucleus.com](mailto:support@atinucleus.com)  
<http://www.atinucleus.com>

Copyright (©) 1999, All Rights Reserved.

Document Part Number : 0001012-002

Last Revised: August 20, 1999





# Contents

Chapter 1 - Introduction .....	1
Introduction .....	2
Basic Information .....	2
Chapter 2 – Getting Starting .....	3
Description of Driver Files .....	4
Installing the Software .....	5
PPP Demo Applications .....	5
Null Modem Applications .....	5
Windows 95: .....	6
Windows NT: .....	7
Running the Demonstration .....	7
Chapter 3 – Implementation and Usage .....	9
NU_Init_Devices Service Call .....	10
Other Nucleus NET Interface Calls .....	10
Nucleus NET in Polled Mode .....	10
PPP Services .....	11
NU_Carrier .....	12
NU_Change_Communication_Mode .....	13
NU_Modem_Control_String .....	14
NU_Terminal_Data_Ready .....	15
NU_Dial_PPP_Server .....	16
NU_Get_Terminal_Char .....	17
NU_PPP_Hangup .....	18
NU_Purge_Terminal_Buffer .....	19
NU_Put_Terminal_Char .....	20
NU_Set_PPP_Login .....	21
NU_Set_PPP_Client_IP_Address .....	22
NU_PPP_Still_Connected .....	23
NU_Wait_For_PPP_Client .....	24
Chapter 4 - Configuration Options .....	25
PPP_MAX_ID_LENGTH and PPP_MAX_PW_LENGTH .....	26
PPP_MAX_HOLDING_PACKETS .....	26
PPP_MAX_HOLDING_PACKETS_PTR .....	26



## Nucleus NET PPP Driver Reference Manual

PPP_MAX_TX_QUEUE_PTRS .....	26
LCP_MAX_ECHO .....	26
LCP_ECHO_VALUE .....	27
LCP_FOREIGN_DEFAULT_ACCM and LCP_LOCAL_DEFAULT_ACCM .....	27
LCP_USE_ACCM .....	27
LCP_DEFAULT_MAGIC_NUMBER .....	27
LCP_DEFAULT_PROTOCOL_COMPRESS .....	27
LCP_DEFAULT_ADDRESS_COMPRESS .....	27
LCP_DEFAULT_AUTH_PROTOCOL .....	28
LCP_MAX_CONFIGURE .....	28
LCP_MAX_TERMINATE .....	28
LCP_MAX_AUTHENTICATE .....	28
LCP_TIMEOUT_VALUE .....	28
LCP_USE_MRU .....	28
MDM_ACCEPT_CALL .....	28
DEBUG_PRINT .....	28
PRINT .....	28
DEBUG_PKT_TRACE .....	28
PKT_TRACE_SIZE .....	29
Link Negotiations .....	29
PPP_MAX_HOLDING_PACKETS and	
PPP_MAX_HOLDING_PACKETS_PTRS .....	32
PPP_MAX_TX_QUEUE_PTRS .....	32
Chapter 5 – PPP Driver Design and Porting .....	33
Porting Issues .....	34
Appendix A –Nucleus PPP Constants .....	35
Nucleus PPP Constants .....	36
Nucleus PPP Constants (Alphabetical Listing) .....	36



1

# Introduction

Introduction  
Basic Information



## Nucleus NET PPP Driver Reference Manual

This document briefly describes the installation of the PPP Driver for Nucleus NET executing on an embedded system. Each version will be specific to the target hardware and may contain differences that are not discussed in this document. For an explanation of items that may be different for your particular target, please refer to the Target Specific Notes that came with your PPP software.

### Introduction

PPP (Point-to-Point Protocol) is a standard for communicating over point-to-point serial lines. Using the PPP driver, Nucleus NET applications can communicate with other architectures (Windows 95, Windows NT, UNIX, etc.) over dial-up serial lines.

The PPP driver is transparent to the application layer, with two exceptions. The first exception is during system initialization, when the COM port, baud rate, etc., must be provided. The second exception is when a dial-up serial link is used, the physical connection must be made before the application attempts to send TCP or UDP packets.

### Basic Information

To make use of the Nucleus NET PPP driver you must also have Nucleus PLUS and Nucleus NET, Accelerated Technology's real-time operating system and TCP/IP protocol stack respectively

**NOTE:** Some portions of this manual may not apply to your particular target (mainly those dealing with modem control functions, some embedded UARTs do not contain modem control functionality).



2

# Getting Started

Description of Driver Files

Installing the Software

PPP Demo Applications



## Description of Driver Files

You should have all of the PPP driver source code, header files, linker control files, and batch files for linking the PPP driver into a Nucleus NET application.

The following is a brief description of the important driver files.

Driver File	Description
PPP.C	This file contains the initialization of PPP, the receive and transmit routines, the routines necessary to control the negotiation phases, and routines for the application.
LCP.C	This file contains the link control protocol and is responsible for establishing and maintaining the link.
CHAP.C	The challenge handshake authentication protocol is contained in this file.
NCP.C	This file contains the network control protocols, specifically the internet protocol control protocol. It is responsible for negotiating which network protocols will be used over the PPP link and configuring the options for each one.
PAP.C	The password authentication protocol is contained in this file.
PW_LIST.C	This file holds the password list that will be used when authenticating a client. New clients and their passwords must be added to this file to allow them access to the system.
URT.C	This file contains the UART specific code. This file is essentially a serial driver for the UART. It is designed to be easily replaced by code for a different UART. Among other things this file initializes the UART and provides character input and output routines.
MDM.C	This file contains the modem portion of the PPP driver. It provides capabilities to dial into a PPP server and to allow a caller to dial into Nucleus NET.
PPP_OPTS.H	This file contains all configuration options for PPP. It currently is set to the default values stated in RFC 1661.
PPP.H	This file includes all necessary headers for an application program to use PPP. It should be included in the user application to provide PPP capabilities.
<demo name>.BAT	This file will assemble the port specific RTOS files, compile the C source file supplied on the command line and link the application together with the RTOS library, network library, and tool libraries.
NET.BAT	This file will build the Nucleus NET library with the correct defines to enable PPP.
PPP.BAT	This batch file will compile the driver files and add them to the Nucleus NET library. It can also be used after the library has been built to update PW_LIST.C and PPP_OPTS.H.



### Installing the Software

The installation of Nucleus PPP varies with the target platform. Please refer to the *Target Specific Notes* that you received for complete installation instructions.

### PPP Demo Applications

New sets of demo applications are shipped with PPP. This is because those shipped with Nucleus NET assume an Ethernet network will be used. The demos used for dialup networking with a standard modem can be found in the directory `DIALUP`. The demos used for null modem communication can be found in the directory `NMODEM`. The main difference between the Nucleus NET applications for Ethernet and those for PPP is that a physical connection must be made with a remote host before network packets can be exchanged.

To build the demos supplied, please refer to the *Target Specific Notes* that you received. Once the demonstrations are built, the following sections describe how to use these demonstration programs.

### Null Modem Applications

Using the null modem applications, a developer can test PPP applications by connecting their device to a standard PC via a null modem cable. The null modem applications were created and tested against a PC executing Windows. They may require modifications for communication with other operating systems.

To execute these applications you will need a Windows machine with which to communicate. You will also have to install dial-up networking. Finally you need to install a modem. Install a standard modem with a baud rate set to an acceptable value for your work.

After both dial-up networking and a standard modem have been installed follow the directions below for the appropriate operating system.



### Windows 95:

Select “*Dial-up Networking*” from the Windows 95 *Start* menu. You will need to make a new connection that uses the standard modem just installed (the phone number is not important). After the new connection has been created, perform the following steps to configure the new connection.

To edit the properties for the new connection:

- Select the “*Configure*” button.

- Select the “*Connection*” tab.

- Select the “*Advanced*” button.

- Disable flow control.

- Back out to the *main property* sheet.

- Now select “*Server Types*”.

- For “*Type of Dial Up Server*”, select “*PPP: Windows 95, Windows NT 3.5, Internet*”.

- Disable “*Log on to network*”.

- Make sure the only allowed network protocol is *TCP/IP*.

- Select the “*TCP/IP Settings*” button.

- Select “*Server assigned IP address*” and “*Server assigned name server addresses*”.

- Disable “*Use IP header compression*”.

- Disable “*Use Default Gateway on Remote Network*”

That completes the configuration of the connection, so select “*OK*” until the property sheet is closed down.



### Windows NT:

Select *"Dial-up Networking"* from the Windows NT *Start* menu. You will need to make a new connection that uses the standard modem just installed (the phone number is not important). After the new connection has been created, perform the following steps to configure the new connection.

To edit the properties for the new connection:

Click the *"More"* button and select *"Edit entry and modem properties"*.

Click the *"Configure"* button.

Disable *"Flow Control"*.

Set the *"Initial speed"* to that of the COM port of the target board. This will be the baudrate used during the PPP session.

Click *"OK"*.

Now select the *"Server"* tab.

For *"Type of Dial Up Server"*, select *"PPP: Windows 95, Windows NT 3.5, Internet"*.

Disable all network protocols except *"TCP/IP"*.

Disable *"Software Compression"* and *"PPP LCP extensions"*.

Select the *"TCP/IP Settings"* button.

Select *"Server assigned IP address"* and *"Server assigned name server addresses"*.

Disable *"Use IP header compression"*.

Disable *"Use Default Gateway on Remote Network"*

That completes the configuration of the connection, so select *"OK"* until the property sheet is closed down.

### Running the Demonstration

You should connect the machine that is running Nucleus NET to the machine that is running Windows, via a null modem cable. Windows does not support PPP over a direct connection. PPP is only supported over a dial-up connection. However, the demo applications for use over a null modem trick Windows into thinking that it is talking to a modem. The following sample code can be found in each of the demo applications. It simply responds with *OK* to any modem commands issued by Windows. When a dial command of ATDT is received, the loop is exited, and Windows thinks that it has successfully dialed up a remote host. From that point on, PPP communications commence.



## Nucleus NET PPP Driver Reference Manual

```
while(1)
{
    /* Receive a MODEM command. */
    DEMO_Get_Modem_String(mstring);

    PRINTF("Received string: %s\n", mstring);

    /* Respond with OK. */
    NU_Modem_Control_String("OK\n\r");

    /* If the command received was a command to dial then Windows
    now thinks that it has a modem connection to a remote HOST.
    Get out of this loop because data exchanged beyond this
    point will be in the form of IP packets. */

    if (strncmp(mstring, "ATDT", 4) == 0)
        break;
}

/* Act like a modem. */
sprintf (mstring, "CONNECT %d\n\r", init_struct.sl_baud_rate);
NU_Modem_Control_String(mstring);
```

The following combinations of the demos are possible:

dial-up TCPCLI	dial-up TCPSERV
dial-up Win95	dial-up TCPSERV
dial-up TCPCLI	null modem TCPSERV
dial-up Win95	null modem TCPSERV
dial-up UDPCLI	dial-up UDPSERV
dial-up Win95	dial-up UDPSERV
dial-up UDPCLI	null modem UDPSERV
dial-up Win95	null modem UDPSERV

If you have problems connecting to the target or starting network communications make sure that all configuration options are set correctly. The main points of interest are: COM port, baud rate, flow control, IP address, HOSTS.C and server name.



# 3

## Implementation and Usage

NU\_Init\_Devices Service Call

Other Nucleus NET Interface Calls

Nucleus NET in Polled Mode

PPP Services



### NU\_Init\_Devices Service Call

The `NU_Init_Devices` service must be called by all Nucleus NET applications before any of the other Nucleus NET services can be used. `NU_Init_Devices` is generally called in the first task to execute. `NU_Init_Devices` should not be called from `Application_Initialize` and it must be called after a call to `NU_Init_NET` has been made. Among other things, this service calls the appropriate initialization routine for a particular device, in the case of PPP this function is `PPP_Initialize`.

The demo applications may have to be modified for use on your system. The areas that will require attention are the IP address, if acting as a server, the COM port used and the baud rate.

A typical call to initialize the PPP Driver looks like the following.

```
devices[0].dv_name = "PPP_Link";
devices[0].dv_init = PPP_Initialize;
devices[0].dv_flags = (DV_POINTTOPOINT | DV_NOARP);
devices[0].dv_hw.uart.com_port      = COM1;
devices[0].dv_hw.uart.baud_rate     = 19200;
devices[0].dv_hw.uart.parity        = PARITY_NONE;
devices[0].dv_hw.uart.stop_bits     = STOP_BITS_1;
devices[0].dv_hw.uart.data_bits     = DATA_BITS_8;

NU_Init_Devices(devices, 1);
```

### Other Nucleus NET Interface Calls

All other interface calls to Nucleus NET (e.g., `NU_Send`, `NU_Recv`) are unaffected by the use of the PPP driver. These services can be used as described in the *Nucleus NET Reference Manual*.

### Nucleus NET in Polled Mode

Nucleus NET supports two modes of operation, polled and interrupt. The *Nucleus NET Target Specific Notes* indicate that building the software without `INTERRUPT` defined will build the polled mode of Nucleus NET. The PPP driver does not support a polled mode. PPP can be used only when interrupts are enabled



### PPP Services

PPP, being a driver level protocol, is for the most part transparent to the application layer. However, before TCP/IP packets can be exchanged via PPP the physical connection must be established. This will typically be done via a dial-up connection. All of the PPP services can be more accurately described as modem services. Most of these services are for driving the modem.

All PPP services take the name of the device as a parameter. This is the same name used in the device initialization structure for each device. By naming the devices and supplying a name to each service the ability to support mutiple PPP links is obtained. The name can be `devices[0].dv_name`, as in the example above, or it can be the literal name.

The following pages list each of the PPP services, and provide aa brief description of each.



### NU\_Carrier

STATUS NU\_Carrier(CHAR \*link\_name)

This function detects the presence of a carrier. Note: this service is only valid on platforms which the UART has the ability to detect the presence of a carrier.

#### Parameters

Parameters	Description
link_name	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Description
NU_TRUE	Indicates that a carrier is present.
NU_FALSE	Indicates that a carrier is not present.
NU_INVALID_LINK	The link name supplied is not a valid PPP device.



### NU\_Change\_Communication\_Mode

```
STATUS NU_Change_Communication_Mode(INT mode, CHAR *link_name)
```

This service switches the PPP driver between network mode and terminal mode. In network mode arriving characters are assumed to be part of a PPP packet and are routed to the PPP ISR. In terminal mode received characters are routed to the modem ISR, which makes them available to the application layer by use of the “terminal” functions, ie. NU\_Get\_Terminal\_Char, NU\_Put\_Terminal\_Char, etc.

#### Parameters

Parameters	Description
mode	The mode of communication desired. The only two valid values are: MDM_NETWORK_COMMUNICATION and MDM_TERMINAL_COMMUNICATION.
link_name	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Descriptions
NU_SUCCESS	Indicates successful completion of the service.
NU_INVALID_MODE	Indicates the mode parameter was not valid.
NU_INVALID_LINK	The link name supplied is not a valid PPP device.



### NU\_Modem\_Control\_String

STATUS NU\_Modem\_Control\_String(CHAR \*string, CHAR \*link\_name)

This service is used to send a control string to the modem. By default a delay of 100 ms is inserted between each character in order to conform with the modem receive logic. If a greater delay is desired a '~' can be inserted into the control string. A delay of ½ second will occur for each '~' in the control string.

#### Parameters

Parameters	Description
string	Pointer to the ASCII string to be sent to the modem.
link_name	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Description
NU_SUCCESS	Indicates successful completion of the service.
NU_INVALID_POINTER	Indicates the string pointer is NULL.
NU_INVALID_LINK	The link name supplied is not a valid PPP device.



### NU\_Terminal\_Data\_Ready

STATUS NU\_Terminal\_Data\_Ready(CHAR \*link\_name)

This service is used to check the terminal mode data buffer for data.

#### Parameters

Parameters	Description
link_name	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Description
NU_TRUE	Indicates that data is present.
NU_FALSE	Indicates that no data is present.
NU_INVALID_LINK	The link name supplied is not a valid PPP device.



### NU\_Dial\_PPP\_Server

```
STATUS NU_Dial_PPP_Server(CHAR *number, CHAR *link_name,  
                          UINT8 *ip_addr)
```

This service commands the modem to dial a phone number and commands PPP to try to establish a link with the PPP server. Before connecting to a PPP server a call to `NU_Set_PPP_Login` must be made in order for a successful connection to be established.

#### Parameters

Parameters	Description
number	Pointer to the phone number to be dialed. NOTE: a '~' can be used in the phone number to insert a ½ second delay.
link_name	Pointer to the name of the PPP link to which this service should be applied.
ip_addr	Pointer to an IP address to be presented to the PPP server for use. The PPP server does not have to allow the client to use this address and in most cases will not allow its' use. The most common use of this parameter is to use the address 0.0.0.0 which requests the PPP server to assign an address to the client.

#### Return Value

Return Values	Description
NU_SUCCESS	Indicates that a connection was made.
NU_INVALID_POINTER	Indicates that the number pointer was NU_NULL.
NU_NO_CONNECT	Indicates that the connection was not made.
NU_NO_CARRIER	Indicates that the receiver did not send a carrier.
NU_BUSY	Indicates that the number dialed was busy.
NU_LCP_FAILED	This means that PPP could not successfully negotiate the options for the link.
NU_LOGIN_FAILED	This means that either the ID or password used for logging into the remote system was wrong. Or that there is no valid account with the remote system.
NU_NCP_FAILED	This means that PPP could not negotiate an IP address.
NU_INVALID_LINK	The link name supplied is not a valid PPP device.



### NU\_Get\_Terminal\_Char

```
STATUS NU_Get_Terminal_Char(CHAR *c, CHAR *link_name)
```

This service retrieves a character from the receive buffer. This service should only be used when in terminal mode. In network mode received characters are being routed to the TCP/IP protocol stack.

#### Parameters

Parameters	Description
c	Pointer to the location to store the received character .
link_name	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Description
NU_SUCCESS	Indicates a character was returned.
NU_NO_DATA	Indicates that no data was available.
NU_INVALID_POINTER	Indicates that a pointer parameter points to NU_NULL.
NU_INVALID_LINK	The link name supplied is not a valid PPP device.



### NU\_PPP\_Hangup

```
STATUS NU_PPP_Hangup(UINT8 mode, CHAR *link_name)
```

This service terminates the PPP link and commands the modem to hang up. Since it is possible that other tasks may be using the network there are two options for terminating the link. The `NU_FORCE` option will close the link and all open sockets on the link, even if the sockets are being used. Any tasks that are suspended on these sockets will be woken up. `NU_NO_FORCE` will only terminate the link if there are not any sockets currently being used on the link.

#### Parameters

Parameters	Description
mode	How to terminate the link. The only two valid values are: <code>NU_FORCE</code> and <code>NU_NO_FORCE</code>
link_name	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Parameters	Description
<code>NU_TRUE</code>	Indicates that the connection has been closed.
<code>NU_FALSE</code>	Indicates that the link could not be closed because the PPP link is currently being used. This will only be returned if the <code>NU_NO_FORCE</code> option is used.
<code>NU_INVALID_LINK</code>	The link name supplied is not a valid PPP device.



### NU\_Purge\_Terminal\_Buffer

STATUS NU\_Purge\_Terminal\_Buffer(CHAR \*link\_name)

This service purges the terminal mode receive buffer of any characters.

#### Parameters

Parameters	Description
link_name	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Description
NU_INVALID_LINK	The link name supplied is not a valid PPP device.
NU_SUCCESS	Indicates the buffer was cleared.



### NU\_Put\_Terminal\_Char

STATUS NU\_Put\_Terminal\_Char(CHAR c, CHAR \*link\_name)

This service puts a character to the serial port. It is used to send characters when in terminal mode.

#### Parameters

Parameters	Description
c	The character to be sent to the serial port.
link_name	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Description
NU_INVALID_LINK	The link name supplied is not a valid PPP device.
NU_SUCCESS	Indicates the character was sent.



**NU\_Set\_PPP\_Login**

```
STATUS NU_Set_PPP_Login (CHAR id[PPP_MAX_ID_LENGTH],
                        CHAR pw[PPP_MAX_PW_LENGTH],
                        CHAR *link_name)
```

This service sets the ID and password to be used when dialing into a service provider. This must be set before an attempt to connect is made, via the `NU_Dial_PPP_Server` service. An ID or PW longer than the preset max will be truncated at the max length, the preset max is set in `PPP_OPTS.H`; see “PPP Configuration Options.” Once the pair is set they will remain set, per link, until another call to `NU_Set_PPP_Login` is made.

**Parameters**

Parameters	Description
id	The id or name to be used when logging on to a remote system.
pw	The password to be used when logging on to a remote system.
link_name	Pointer to the name of the PPP link to which this service should be applied.

**Return Value**

Return Values	Description
NU_INVALID_LINK	The link name supplied is not a valid PPP device.
NU_SUCCESS	Indicates that the ID and PW were set.



### NU\_Set\_PPP\_Client\_IP\_Address

```
STATUS NU_Set_PPP_Client_IP_Address(UINT8 ip_address,  
                                     CHAR *link_name)
```

This service sets the IP address to be assigned to a calling client for a particular PPP link. This is only used when a call to `NU_Wait_For_PPP_Client` is made. Once the IP address is set it will remain set, per link, until another call to `NU_Set_PPP_Client_IP_Address` is made.

#### Parameters

Parameters	Description
<code>ip_address</code>	Pointer to the IP address to be assigned to a calling client for for the supplied link.
<code>link_name</code>	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Description
<code>NU_SUCCESS</code>	Indicates that the address was taken.
<code>NU_INVALID_ADDRESS</code>	Indicates that the address to be assigned is the same as the one NET is using.
<code>NU_INVALID_LINK</code>	The link name supplied is not a valid PPP device.



## NU\_PPP\_Still\_Connected

STATUS NU\_PPP\_Still\_Connected(CHAR \*link\_name)

This service is similar to `NU_Carrier`, but instead of returning the state of the modem it returns the logical state of the link.

### Parameters

Parameters	Description
link_name	Pointer to the name of the PPP link to which this service should be applied.

### Return Value

Return Values	Description
NU_TRUE	Indicates that the link is still open.
NU_FALSE	Indicates that the link is closed.
NU_INVALID_LINK	The link name supplied is not a valid PPP device.

### NU\_Wait\_For\_PPP\_Client

```
STATUS NU_Wait_For_PPP_Client(UINT8 *server_ip_address,  
                              CHAR *link_name)
```

This service puts the modem in answer mode and waits for a client to call. Once the modems have connected, PPP will start negotiating the link and authenticating the client. When control returns from this function a client has successfully negotiated the PPP link and is currently connected to the system, assuming the returned status is `NU_SUCCESS`. Once a client starts a PPP session the IP address are added to Nucleus NET's routing table. Note that a call to `NU_Set_PPP_Client_IP_Address` must be made before a client can be connected.

#### Parameters

Parameters	Description
<code>server_ip_address</code>	The IP address to be used by Nucleus NET for the local end of the PPP link.
<code>link_name</code>	Pointer to the name of the PPP link to which this service should be applied.

#### Return Value

Return Values	Description
<code>NU_INVALID_LINK</code>	The link name supplied is not a valid PPP device.
<code>NU_SUCCESS</code>	A client has successfully connected.



4

## Configuration Options



There are a number of configuration options available for PPP. All of these can be found in the file `PPP_OPTS.H`. The options shipped by default are the default values specified in RFC 1661, where applicable, and should allow connection to any service provider. They are made available for user configuration. All options and their valid settings are listed below. The use for some of these options maybe confusing. Please see the diagram at the end of this section for more clarification on how they are used, specifically for the options `PPP_MAX_HOLDING_PACKETS`, `PPP_MAX_HOLDING_PACKETS_PTR`, and `PPP_MAX_TX_QUEUE_PTRS`.

### **PPP\_MAX\_ID\_LENGTH and PPP\_MAX\_PW\_LENGTH**

These specify the maximum number of characters an ID and PW can be. This applies to the ID and PW used when dialing out and when a client is dialing in.

### **PPP\_MAX\_HOLDING\_PACKETS**

PPP uses a small ring buffer to hold pointers to incoming packets for passing the packets from the LISR to the HISR. This option defines how big the ring buffer will be. A buffer that is too small may result in packets being lost.

### **PPP\_MAX\_HOLDING\_PACKETS\_PTR**

This defines the number of pointers in a ring buffer used to pass completed packets from the RX LISR to the PPP HISR. These pointers point to the buffers mentioned above. The more UARTs that are being used the greater this value should be. Example: if the above macro is set to 3, there will be 3 buffers for each UART. If two UARTs are being used then there will be 6 total RX buffers. There should be enough pointers to pass those buffers to the RX hisr. Therefore this value should be 6 to handle the worst case situation.

### **PPP\_MAX\_TX\_QUEUE\_PTRS**

After a packet has been completely transmitted the device that completed the transmission is passed to the TX HISR. This is done so that the TX HISR can deallocate the buffers used by the packet and so it can send the next packet that is in queue, if there is one. This macro defines how many pointers are available to pass the device to the TX HISR.

### **LCP\_MAX\_ECHO**

LCP echo request packets are used by PPP to determine if the link has been lost. At a time interval specified by `LCP_ECHO_VALUE`, PPP sends an echo request and expects to receive an echo reply. This macro is the number of echo request packets that will be sent without a reply before the link is considered to be lost.



### **LCP\_ECHO\_VALUE**

This is the time between echo request packets. It is in clock ticks and is relative to the clock speed of the system. The time between requests should be long enough to allow for the reply to return.

### **LCP\_FOREIGN\_DEFAULT\_ACCM and LCP\_LOCAL\_DEFAULT\_ACCM**

This is the default asynchronous control character map. This specifies which characters, when transmitted, should be made transparent to the hardware. This is a 32-bit map and each bit corresponds to the ASCII equivalent character. Since some hardware devices use control characters this map gives a way to hide the control characters from the hardware. If it is known exactly what control characters may effect the specific hardware used then this can be set to those characters. If it is not known then the default value should be used to mask all ASCII characters below 32. There are default values for both the foreign and local ends of the PPP link. The local ACCM can be changed to fit any particular hardware being used. In most cases, if the hardware does not respond to control codes on the line, this value can be set to 0x0. The foreign ACCM default value should be left as 0xffffffff. During LCP negotiation the foreign host will inform the local end of the link what its' ACCM should be set to.

### **LCP\_USE\_ACCM**

This is a boolean option specifying `NU_TRUE` or `NU_FALSE` depending on if the use of the ACCM is desired. A value of `NU_TRUE` is suggested for this option.

### **LCP\_DEFAULT\_MAGIC\_NUMBER**

This is a boolean variable and defines whether or not the magic number option will be used. Valid values for this option are `NU_TRUE` and `NU_FALSE`. The function of the magic number option is to detect if the dial-up line is looped back or not.

### **LCP\_DEFAULT\_PROTOCOL\_COMPRESS**

This is a boolean variable and defines whether or not protocol field compression will be used. Valid values for this option are `NU_TRUE` and `NU_FALSE`. Using this compression will save 1 byte per packet sent/received.

### **LCP\_DEFAULT\_ADDRESS\_COMPRESS**

This is a boolean variable and defines whether address and control field compression will be used. Valid values for this option are `NU_TRUE` and `NU_FALSE`. Using this compression will save 2 bytes per packet sent/received.



## Nucleus NET PPP Driver Reference Manual

### **LCP\_DEFAULT\_AUTH\_PROTOCOL**

This tells PPP which authentication protocol to try first when authenticating a calling client. Valid values for this are 0xC223 for CHAP and 0xC023 for PAP.

### **LCP\_MAX\_CONFIGURE**

Maximum number of times to retry configuration of the link. This will apply to LCP and NCP negotiation.

### **LCP\_MAX\_TERMINATE**

Maximum number of times to retry terminating the link.

### **LCP\_MAX\_AUTHENTICATE**

Maximum number of times to retry authentication.

### **LCP\_TIMEOUT\_VALUE**

This is the time between retransmission of negotiation and authentication packets. It is in clock ticks and is relative to the speed of the hardware.

### **LCP\_USE\_MRU**

This turns the minimum receive unit option off/on. This option should only be used if the MRU for the link has been changed from the default value of 1500 bytes. The MRU to be used for the link can be changed in the file `PPP_DEFS.H` by modifying the macro `PPP_MTU`.

### **MDM\_ACCEPT\_CALL**

This define is used to put the modem into answer mode. It will control the number of rings needed before the modem will answer the phone.

### **DEBUG\_PRINT**

Enables printing of negotiating information.

### **PRINT**

Defines where the printing of debug information will be sent.

### **DEBUG\_PKT\_TRACE**

Enables the capturing of all packets sent and received.



### **PKT\_TRACE\_SIZE**

Defines the size of the packet trace buffer. When the buffer is filled it will wrap to the beginning and continue to log all packets.

**NOTE:** All default values listed above are only used during PPP initialization of the link. The exception to this is `LCP_FOREIGN_DEFAULT_ACCM`, it is used every time a PPP session is started. After initialization these values can be changed by use of the PPP service `NU_Set_PPP_Link_Options`.

### **Link Negotiations**

PPP has been designed to negotiate all configuration options automatically without help from the user. Sometimes this negotiation fails for seemingly no reason. PPP does not guarantee that a successful connection will be made every time a connection is attempted. If connection fails try to make the connection again. If after several attempts connection is still not made there may be a configuration option that the service provider does not understand and is causing the negotiation to fail. Contact with the service provider may be necessary in order to match the configuration options specified in `PPP_OPTS.H` with those specified by the SP.



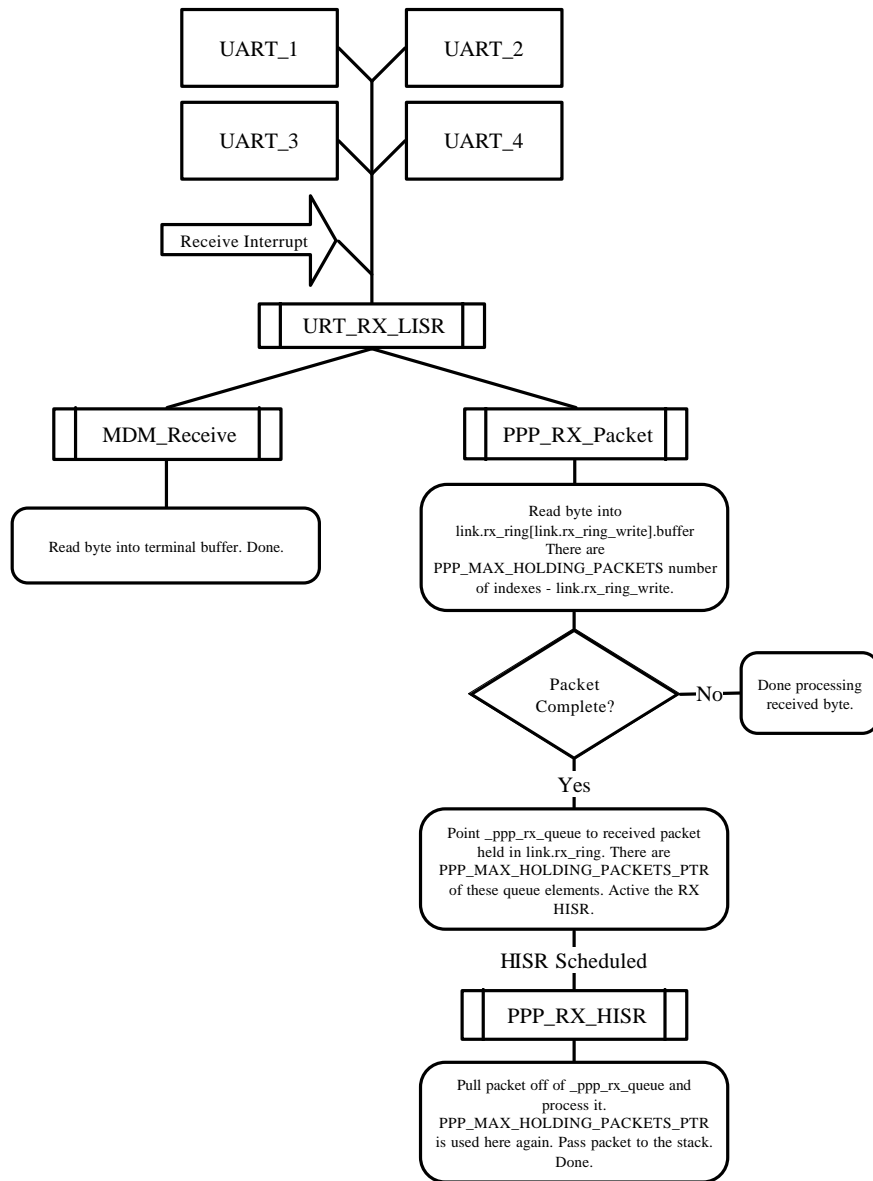


Figure 1 - PPP\_MAX\_HOLDING\_PACKETS and PPP\_MAX\_HOLDING\_PACKETS\_PTRS

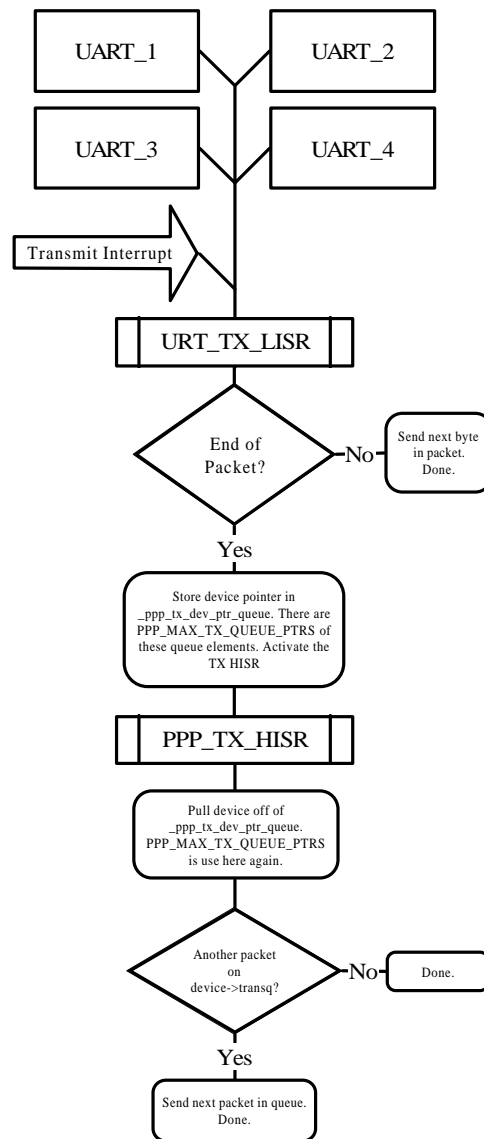


Figure 2 - PPP\_MAX\_TX\_QUEUE\_PTRS

### **PPP\_MAX\_HOLDING\_PACKETS and PPP\_MAX\_HOLDING\_PACKETS\_PTRS**

PPP\_MAX\_HOLDING\_PACKETS is used to declare an array that will hold the packet that is currently being received. Since PPP packets are received a byte at a time there needs to be buffers in place to hold the packet until it has been completely received, these buffers are `link.rx_ring`; a ring of buffers for each PPP link. From Figure 1, it can be seen that once the packet has been fully received it is passed to the RX HISR by means of a second array. This array is of size PPP\_MAX\_HOLDING\_PACKETS\_PTRS. It consists only of pointers that point to the packets in the `link.rx_ring`. So when the RX HISR is processing a packet the data it is processing is still on a link's receive ring buffer. Therefore, this ring buffer needs to be large enough to hold already received packets and still have room to receive more packets until the RX HISR is done processing previously received ones. Since the RX HISR is responsible for processing packets for all UARTs its' pointer array, `_ppp_rx_queue`, needs to be large enough to point to received packets from multiple UARTs, if more than one UARTs is being used. A general rule for the worst case situation would be three PPP\_MAX\_HOLDING\_PACKET per UART and the total of that for PPP\_MAX\_HOLDING\_PACKETS\_PTRS. Since PPP\_MAX\_HOLDING\_PACKETS\_PTRS is only an array of pointers, this is not that expensive of a configuration. PPP\_MAX\_HOLDING\_PACKET, on the other hand, is an array of PPP frames. Since a PPP frame can be no smaller than 1500 bytes, this option can consume large amounts of memory if there is a significant number of UARTs being used.

### **PPP\_MAX\_TX\_QUEUE\_PTRS**

This macro is used to declare an array that holds pointers to PPP device structures. When the UART code has completed transmission of a PPP packet it will put the address of the PPP device that has completed transmission in this queue. Then it will active the TX HISR, please see Figure 2. The TX HISR pulls the device pointer out of this queue and if there is another packet ready for transmission the HISR will start its' transmission. For a worst case situation the size of this macro should be large enough to hold two device pointers from each PPP device in the system.



5

# PPP Driver Design and Porting

Porting Issues



### Porting Issues

Most of the components that make up PPP are independent of the hardware. When porting, attention only needs to be given to one part of PPP, the UART driver. All of code that deals with this can be found in the files `URT.C`, `URT_DEFS.H` and `URT_EXTR.H`. For PPP to function the functions supplied in `URT.C` must be modified to work with the new hardware.

These functions are:

```
URT_Get_Char  
URT_Init_Port  
URT_LISR  
URT_Port_To_Vector  
URT_Put_Char  
URT_Set_Baud_Rate
```

Some UARTs have special registers for monitoring and controlling a modem. If this is the case, `UART_Carrier` should also be modified to fit the hardware.

If the UART being used does not contain the above-mentioned registers, `UART_Carrier` should be stubbed out. In the case of a return value, `NU_TRUE`, should be returned.





# Appendix

## Nucleus PPP Constants



### Nucleus PPP Constants

This appendix contains all of the Nucleus PPP Constants. The following table lists all definitions used by Nucleus PPP:

#### Nucleus PPP Constants (Alphabetical Listing)

Name	Decimal Value	Hex Value
NU_BUSY	-75	FFFFFFB5
NU_FALSE	0	0
NU_INVALID_ADDRESS	-37	FFFFFFD8
NU_INVALID_LINK	-76	FFFFFFB4
NU_INVALID_MODE	-100	FFFFFF9C
NU_INVALID_POINTER	-15	FFFFFFF1
NU_LCP_FAILED	-5	FFFFFFFB
NU_LOGIN_FAILED	-7	FFFFFFF9
NU_NCP_FAILED	-10	FFFFFFF6
NU_NO_CARRIER	-50	FFFFFFCE
NU_NO_CONNECT	-101	FFFFFF9B
NU_NO_DATA	-48	FFFFFFD0
NU_SUCCESS	0	0
NU_TRUE	1	1

